



# A1X Android4.0 3G Dongle 调试说明文档

V1.0  
2012-7-12



## Revision History

Version	Date	Section/ Page	Changes compared to previous issue
1.0	20120712		initialition



# 目录

目录.....	3
1. 基本概念.....	4
2. 制作 VID-PID 文件 .....	4
<b>3.1 方法一：利用 PC 设备管理 .....</b>	<b>6</b>
3.2.2. 查看 DefaultVendor、DefaultProduct.....	6
3.2.2. 查看 TargetVendor、 TargetProduct .....	7
3.2.2. 读取 MessageContent.....	9
<b>3.2 方法二：利用 USB 分析仪.....</b>	<b>15</b>
3.2.2. 抓取数据 .....	15
3.2.2. 分析数据 .....	15
3.2.2.1. 转换前的 vid 和 pid: .....	15
3.2.2.2. 转换后的 vid 和 pid: .....	16
3.2.2.3. MessageContent:.....	16
3. 确定 AT、PPP 口.....	17
4. 修改内核层.....	21
5. 修改 android 层 .....	22
6. 常见问题.....	22
7. Declaration .....	23

## 1. 基本概念

USB 3G Dongle 有些要切换，有些不要切换。不需要切换的 3G Dongle 一接上电脑就有串口设备(若没装驱动 Windows 上则表现为新的黄色感叹号设备)。切换的 USB 3G Dongle 分两种模式，USB Mass Storage Device 模式和串口模式。从 USB Mass Storage Device 模式切换到串口模式，需要发送指令到 USB CD-ROM 设备，USB CD-ROM 设备接收到该指令后该 3G Dongle 就会断开连接然后重新连接到主机，这时候枚举就会有串口和 Modem 口出现了，最终要上网需要借助串口和 Modem 口。用于切换的指令一般是由该 Dongle 的应用程序来发送的，该指令是嵌入到下面要抓的 MessageContent 发送的。

## 2. 制作 VID-PID 文件

vid\_pid 文件在目录 usb\_modeswitch.d 下,以华为的 12d1\_1003 和 12d1\_1031 为例，内容如下：

示例一：12d1\_1003 文件内容：

```
#####  
# Huawei E220, E230, E270, E870  
↵  
DefaultVendor= 0x12d1  
DefaultProduct=0x1003  
↵  
TargetClass=0xff  
↵  
CheckSuccess=20  
↵  
HuaweiMode=1
```

示例二：12d1\_1031 文件内容：

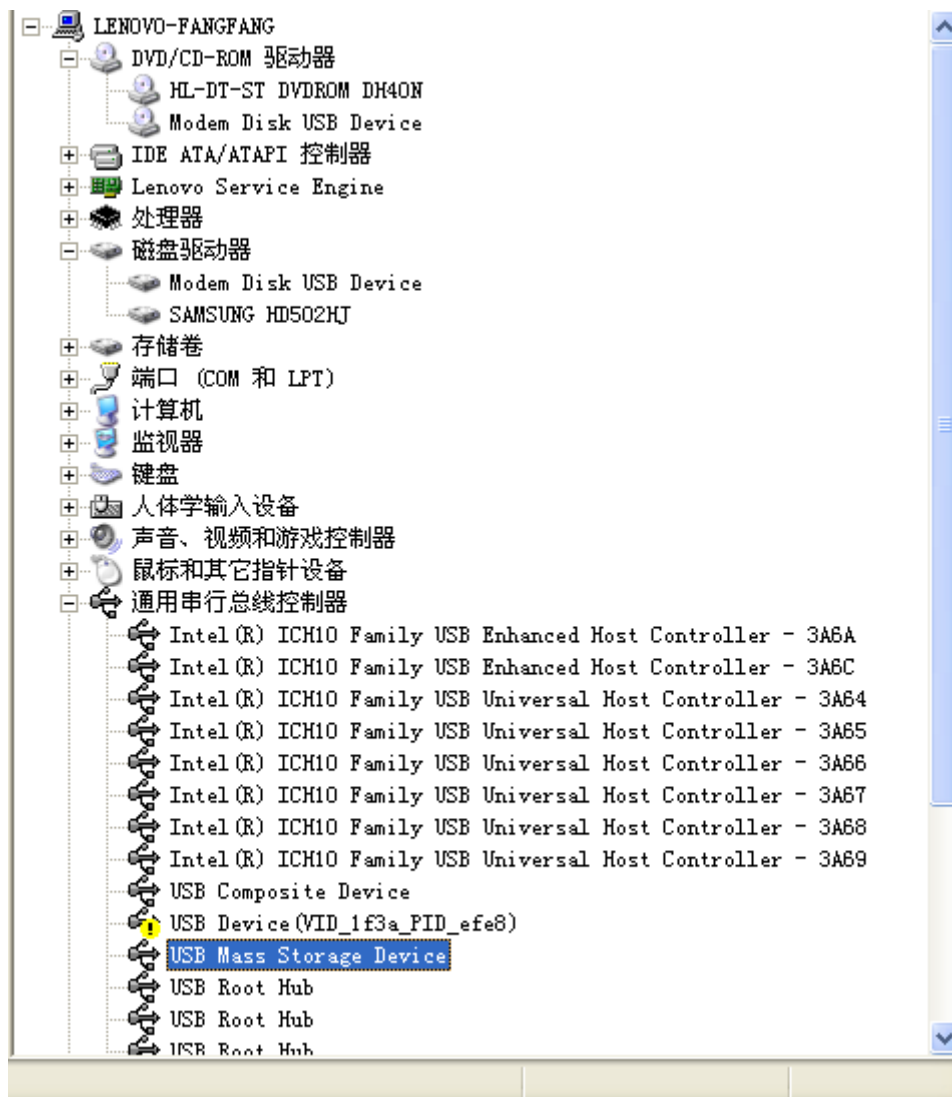


### 3.1 方法一：利用 PC 设备管理

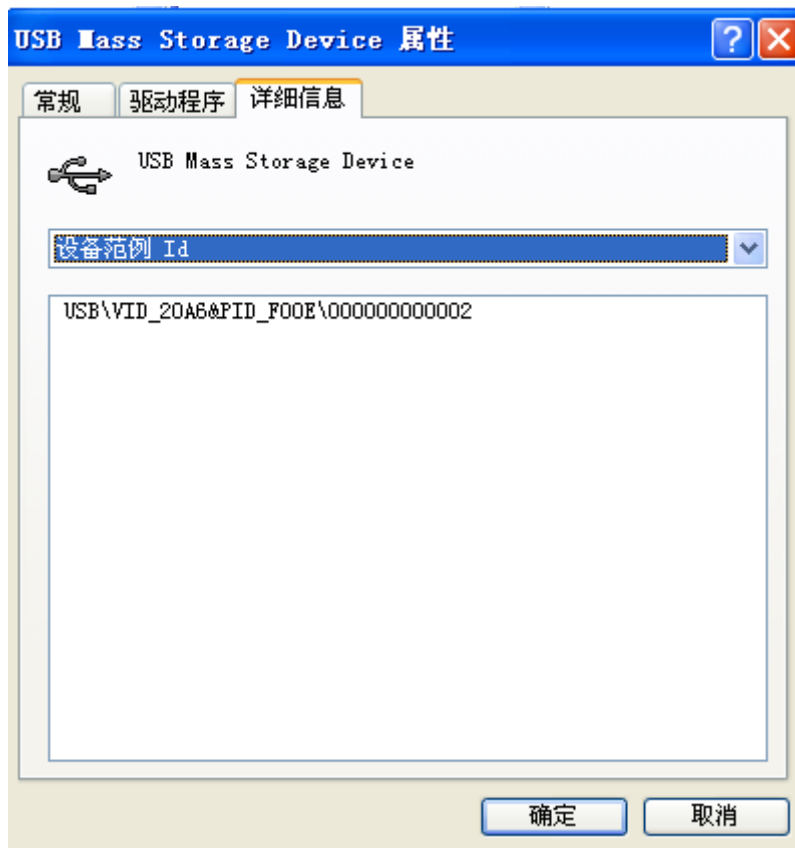
这里借助 Windows 系统的设备管理器来得到 vid、pid。具体步骤如下。

#### 3.2.2. 查看 DefaultVendor、DefaultProduct

先不要装该 Dongle 的驱动程序，打开“设备管理器”，插入 3G Dongle，等“设备管理器”出现新设备后，如下图所示：



出现上图后，右键单击“属性”，然后选择“详细信息”，将会看到下面图所示：



上图 USB\VID\_20A6&PID\_F00E\000000000002 中的 VID\_20A6&PID\_F00E 表示该设备的 vid=0x20a6, pid=0xf00e. 这里就是 DefaultVendor, DefaultProduct。即：

**DefaultVendor=0x20a6**

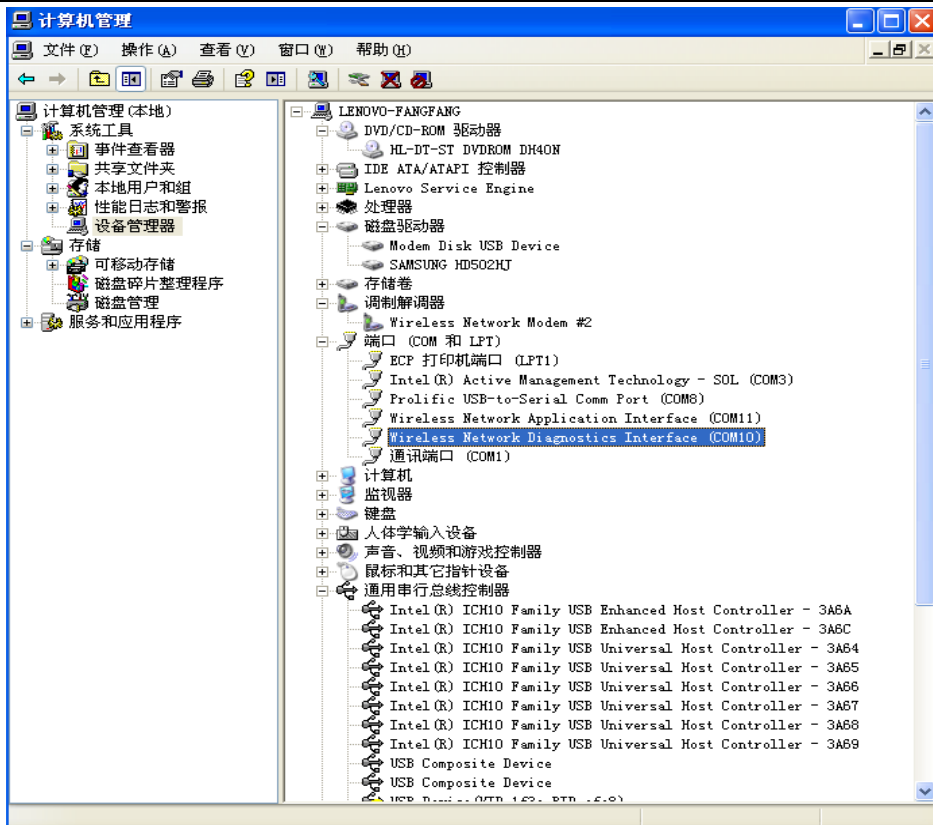
**DefaultProduct=0xf00e**

文件名即为：**20a6\_f00e**

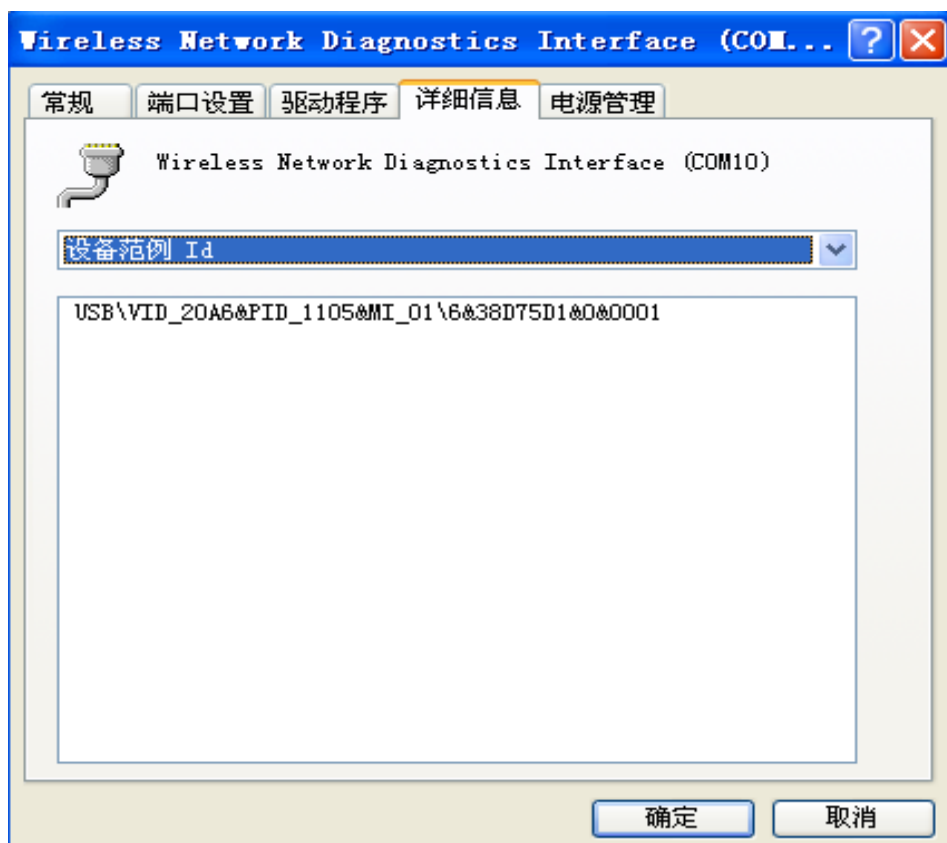
### 3.2.2. 查看 TargetVendor、 TargetProduct

得到 DefaultVendor, DefaultProduct 后, 在 PC 机上安装 USB CD-ROM 中自带的驱动程序了, 如何安装请参考 USB CD-ROM 中自带的说明文档。 驱动程序安装完后, 打开该驱动程序, 这个时候 USB 3G Dongle 会切换成另外一个组合设备, 该组合设备有几个串口和一个或者几个 USB Mass Storage 设备。

打开驱动程序后, 设备管理中会出现新的串口和 Modem 口, 如下图所示:



右击点击“属性”得到 vid 和 pid，如下图所示：





由上图，我们得到了转换后的串口设备的 vid 和 pid， 另外一个串口以及 Modem 口的 vid,pid 一般都一样的， 所以看一个串口设备的 vid, pid 就可以了。即：

**TargetVendor = 0x20A6**

**TargetProduct = 0x1105**

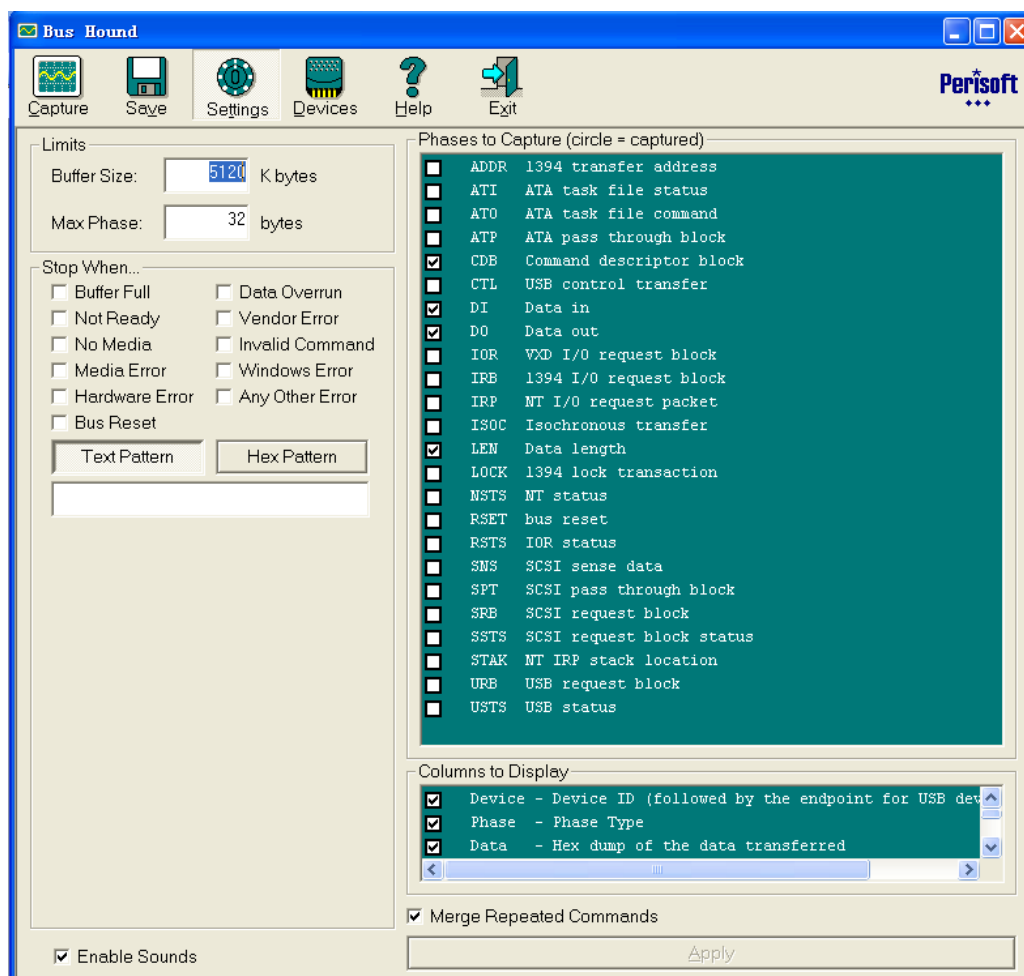
### 3.2.2. 读取 MessageContent

读取 MessageContent 需要借助 Bus Hound, 可以用 bushound5.0 完美版。这里用 BusHound 来抓取 USB 设备的数据流。

#### ● 设置 bushound

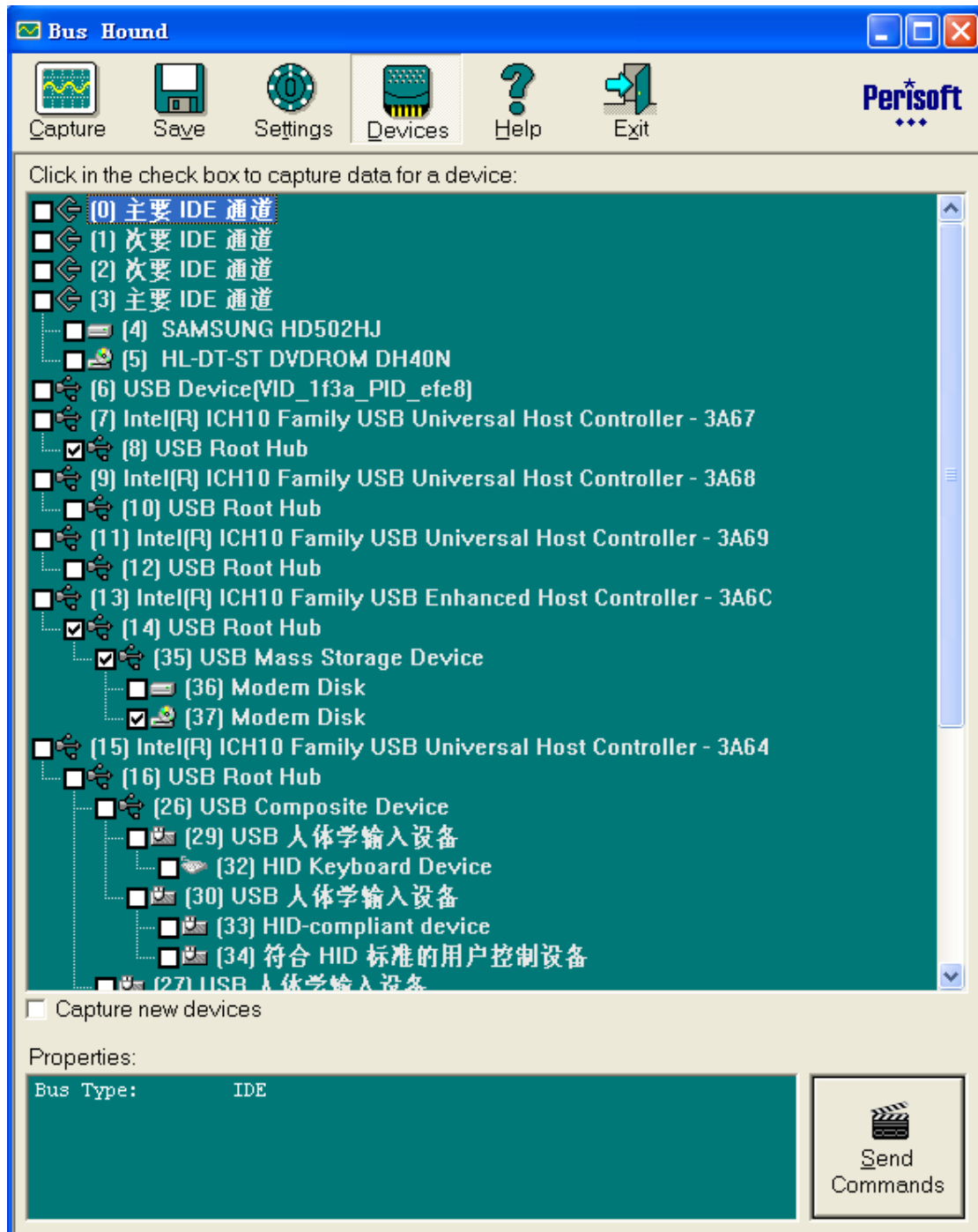
设置 bushound 前，先插入 USB 3G Dongle.因为下面需要选择 USB 设备。这时候不要插入其它 USB 设备。

##### 1. 设置 Settings



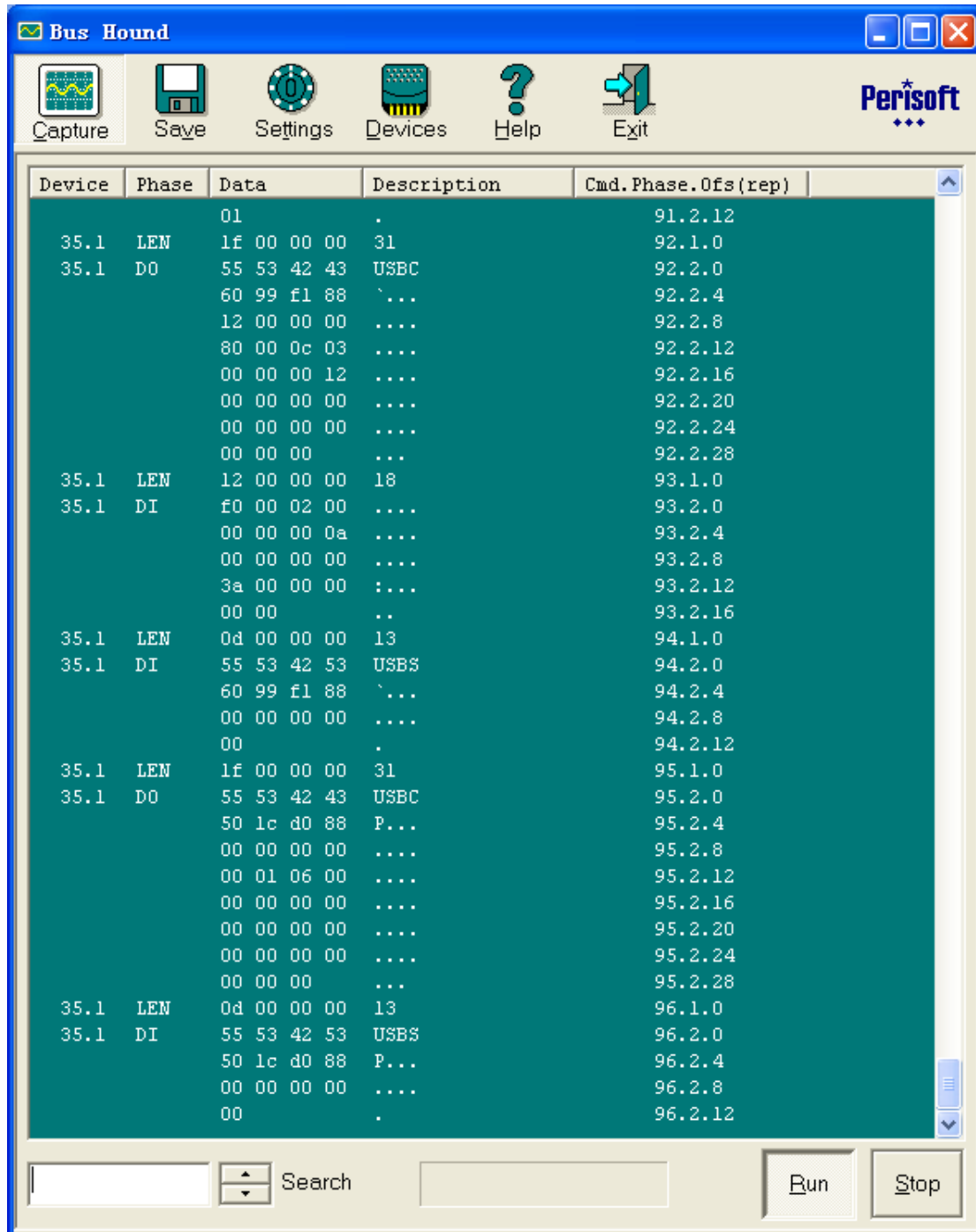
## 2.设置 “Devices”

这里选择 USB 3G Dongle 相关的 USB Mass Storage 设备。注意这里一定要选对设备，且只能选择该 3G Dongle 的 Mass Storage 设备，不要选其它 Dongle 设备，也不要选 USB 串口设备。这一步必须保证 Dongle 的驱动程序还没有打开，相应的串口设备也还没转换出来。



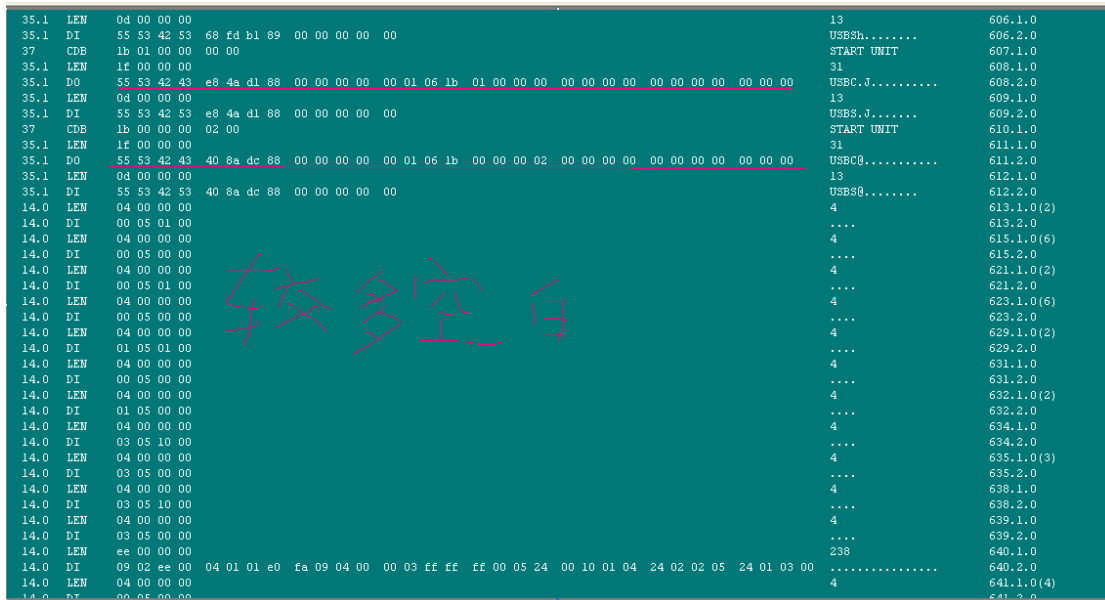
## 2. Capture

点击“Capture”，然后，点击“Run”，如下图所示：



## 4. 抓取 MessageContent

打开 USB 3G Dongle 应用程序，等你看到有下图所示的较多空白时候，点击”Stop”停止抓图，然后点击 “save” 保存抓到的 log. 这时候切换的数据已经抓取到了，也就是 MessageContent 的内容，抓图如下：



**注意：** 如果没有较多空白也没关系，你发现串口设备出来了就可以停止抓数据了，只不过后面找 MessageContent 要较长时间。这时候参考下一节如何分析 bushound log 数据来提取 MessageContent，上图红线所示就是要抓的 MessageContent. 两个：

MessageContent="55534243e84ad1880000000000001061b01000000000000000000000000000000"

MessageContent="55534243e84ad1880000000000001061b00000000000000000000000000000000"

上面两个 MessageContent 建议都试下，有些 3G 设备两个都可以用，有些只能用其中的一个。所以 vid\_pid 文件也就完成了，如下：

```
#####
# example
DefaultVendor= 0x20a6
DefaultProduct= 0xf00e
TargetVendor= 0x20a6
TargetProduct= 0x1103
MessageContent="55534243e84ad1880000000000001061b00000000000000000000000000000000"
CheckSuccess=20
```



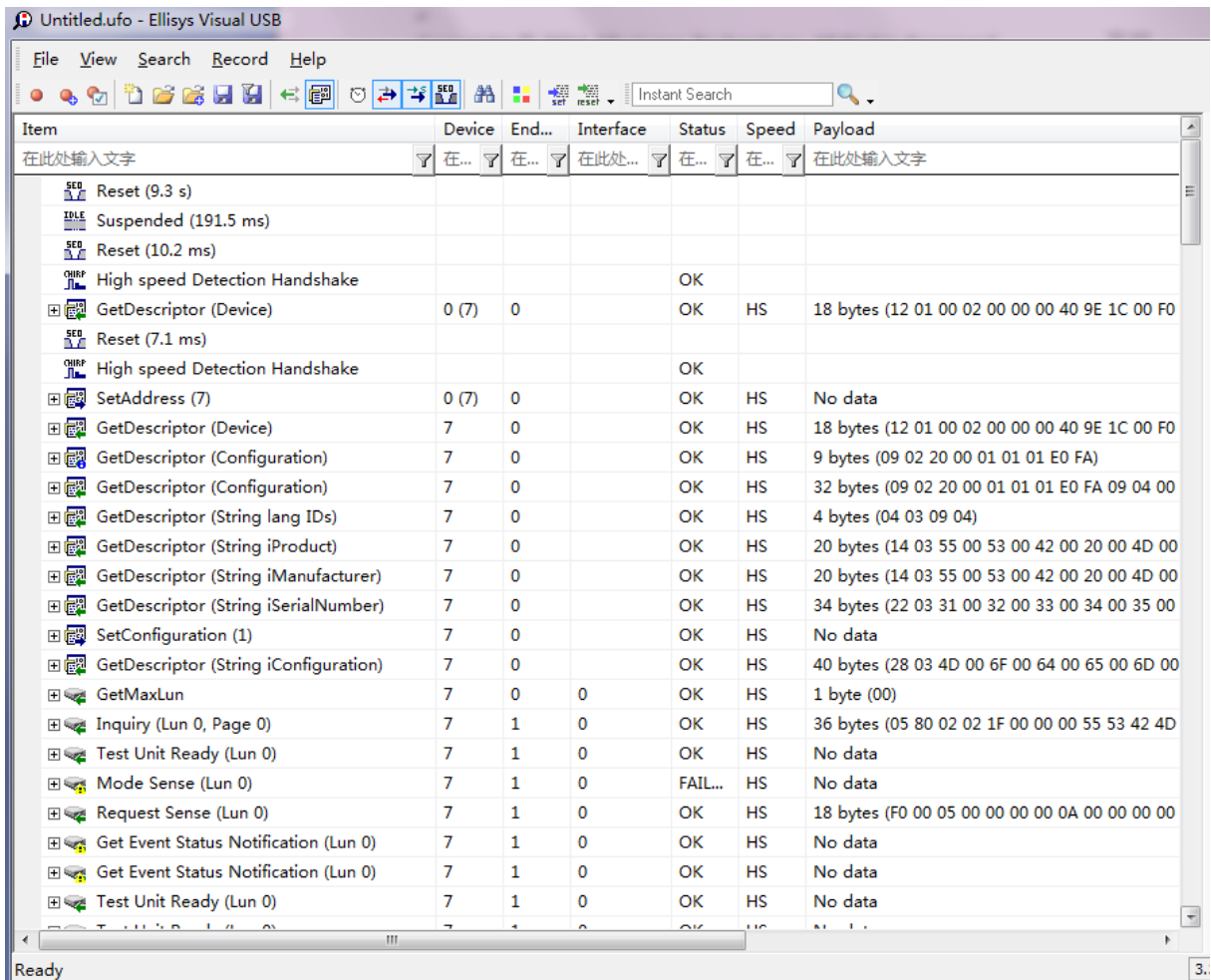


## 3.2 方法二：利用 USB 分析仪

利用 USB 分析仪可以较为准确的得到转换前后的 vid 和 pid, 以及 MessageContent 内容, 本节讲述内容利用 USB 分析仪获取数据, 步骤如下:

### 3.2.2. 抓取数据

在 PC 中安装好 3G Dongle 自带的驱动, 把 USB 分析仪与 PC 机连接好, 打开 3G Dongle 驱动程序, 此时把 3G Dongle 插入 USB 分析仪中, 抓去数据如下图所示:



Item	Device	End...	Interface	Status	Speed	Payload
Reset (9.3 s)						
Suspended (191.5 ms)						
Reset (10.2 ms)						
High speed Detection Handshake				OK		
GetDescriptor (Device)	0 (7)	0		OK	HS	18 bytes (12 01 00 02 00 00 00 40 9E 1C 00 F0)
Reset (7.1 ms)						
High speed Detection Handshake				OK		
SetAddress (7)	0 (7)	0		OK	HS	No data
GetDescriptor (Device)	7	0		OK	HS	18 bytes (12 01 00 02 00 00 00 40 9E 1C 00 F0)
GetDescriptor (Configuration)	7	0		OK	HS	9 bytes (09 02 20 00 01 01 01 E0 FA)
GetDescriptor (Configuration)	7	0		OK	HS	32 bytes (09 02 20 00 01 01 01 E0 FA 09 04 00)
GetDescriptor (String lang IDs)	7	0		OK	HS	4 bytes (04 03 09 04)
GetDescriptor (String iProduct)	7	0		OK	HS	20 bytes (14 03 55 00 53 00 42 00 20 00 4D 00)
GetDescriptor (String iManufacturer)	7	0		OK	HS	20 bytes (14 03 55 00 53 00 42 00 20 00 4D 00)
GetDescriptor (String iSerialNumber)	7	0		OK	HS	34 bytes (22 03 31 00 32 00 33 00 34 00 35 00)
SetConfiguration (1)	7	0		OK	HS	No data
GetDescriptor (String iConfiguration)	7	0		OK	HS	40 bytes (28 03 4D 00 6F 00 64 00 65 00 6D 00)
GetMaxLun	7	0	0	OK	HS	1 byte (00)
Inquiry (Lun 0, Page 0)	7	1	0	OK	HS	36 bytes (05 80 02 02 1F 00 00 00 55 53 42 4D)
Test Unit Ready (Lun 0)	7	1	0	OK	HS	No data
Mode Sense (Lun 0)	7	1	0	FAIL...	HS	No data
Request Sense (Lun 0)	7	1	0	OK	HS	18 bytes (F0 00 05 00 00 00 00 0A 00 00 00 00)
Get Event Status Notification (Lun 0)	7	1	0	OK	HS	No data
Get Event Status Notification (Lun 0)	7	1	0	OK	HS	No data
Test Unit Ready (Lun 0)	7	1	0	OK	HS	No data

### 3.2.2. 分析数据

#### 3.2.2.1. 转换前的 vid 和 pid:

点击设备转换之前的 GetDescriptor(Device), 详细信息栏显示如下图所示:



### GetDescriptor (Device)

Device descriptor	
bcdUSB	2.0
bDeviceClass	Class defined at interface level
bDeviceProtocol	None
bMaxPacketSize0	64
idVendor	0x1C9E
idProduct	0xF000
bcdDevice	0.0
iManufacturer	3 "USB Modem"
iProduct	2 "USB Modem"
iSerialNumber	4 "1234567890ABCDEF"

由上图可知 DefaultVendor = 0x1c9e, DefaultProduct = 0xf000;

#### 3.2.2.2. 转换后的 vid 和 pid:

点击设备转换之后的 GetDescriptor(Device), 详细信息栏显示如下图所示:

### GetDescriptor (Device)

Device descriptor	
bcdUSB	2.0
bDeviceClass	Class defined at interface level
bDeviceProtocol	None
bMaxPacketSize0	64
idVendor	0x1C9E
idProduct	0x9605
bcdDevice	0.0
iManufacturer	2 "USB Modem"
iProduct	1 "Modem Configuration"
iSerialNumber	3 "1234567890ABCDEF"

由上图可知: TargetVendor = 0x1c9e, TargetProduct = 0x9605;

#### 3.2.2.3. MessageContent:

点击转换成功之前的 UnknownCommand,得到 MessageContent, 如下图所示:



Unknown Command 0x06 (Lun...	7	1	OK	HS	No data
Command Transport	7	1	OK	HS	31 bytes (55 53 42 43 12 34 56 78 80 00 00 00 80 00 06 06 F5 04 02 52 70 00 00 00 00 00 00 00)
Data Transport	7	0	OK	HS	No data
Invalid packet	?	?	INV...	HS	
Reset (98.6 ms)					
High speed Detection Handsh...			TIM...		
Suspended (168.6 ms)					
Reset (10.2 ms)					
High speed Detection Handsh...			OK		
GetDescriptor (Device)	0 (8)	0	OK	HS	18 bytes (12 01 00 02 00 00 00 40 9E 1C 05 96 00 00 02 01 03 01)
Reset (7.2 ms)					
High speed Detection Handsh...			OK		
SetAddress (8)	0 (8)	0	OK	HS	No data
GetDescriptor (Device)	8	0	OK	HS	18 bytes (12 01 00 02 00 00 00 40 9E 1C 05 96 00 00 02 01 03 01)

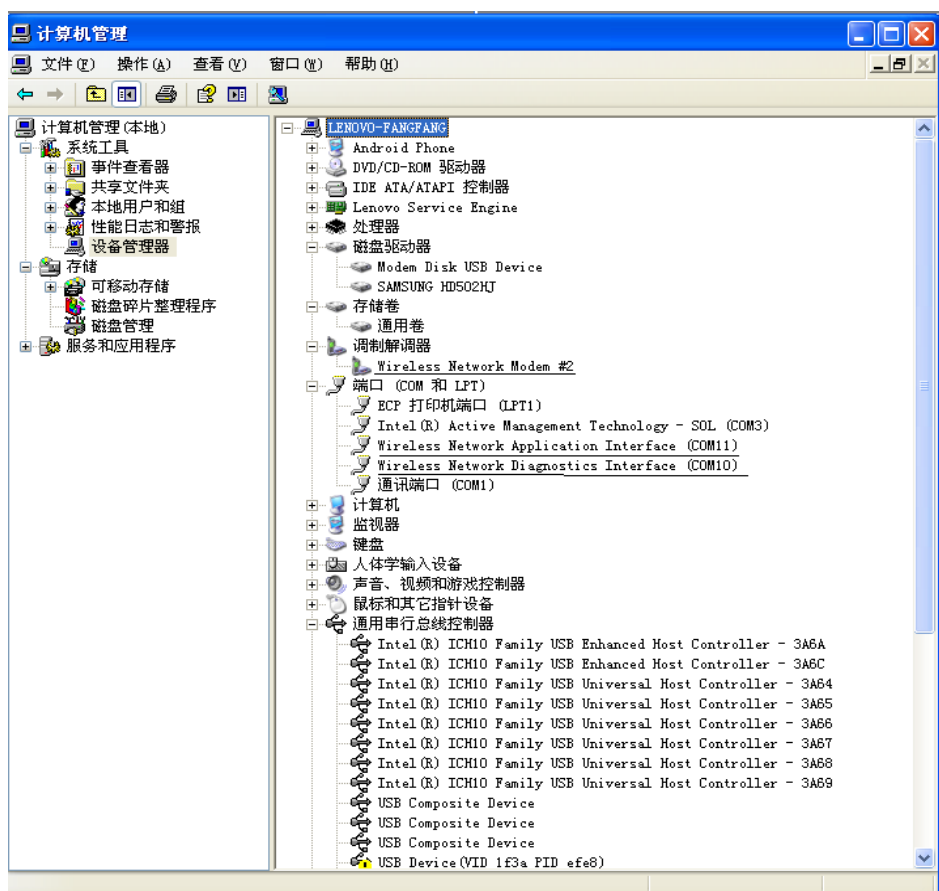
有图可知:

MessageContent = 55 53 42 43 12 34 56 78 80 00 00 00 80 00 06 06 F5 04 02 52 70 00 00 00 00 00 00 00 00

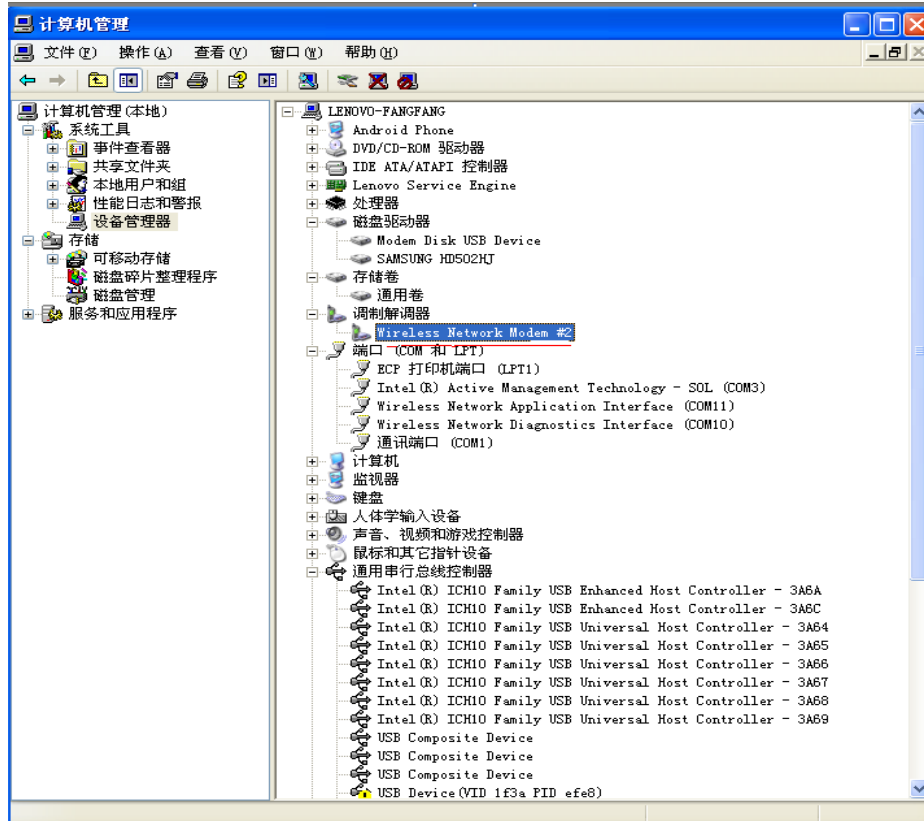
### 3. 确定 AT、PPP 口

在 Android 环境下需要知道哪个 ttyUSBx 是 AT 口，哪个是 PPP 口(即 Modem 口)。这里还是在 Windows 环境来进行。

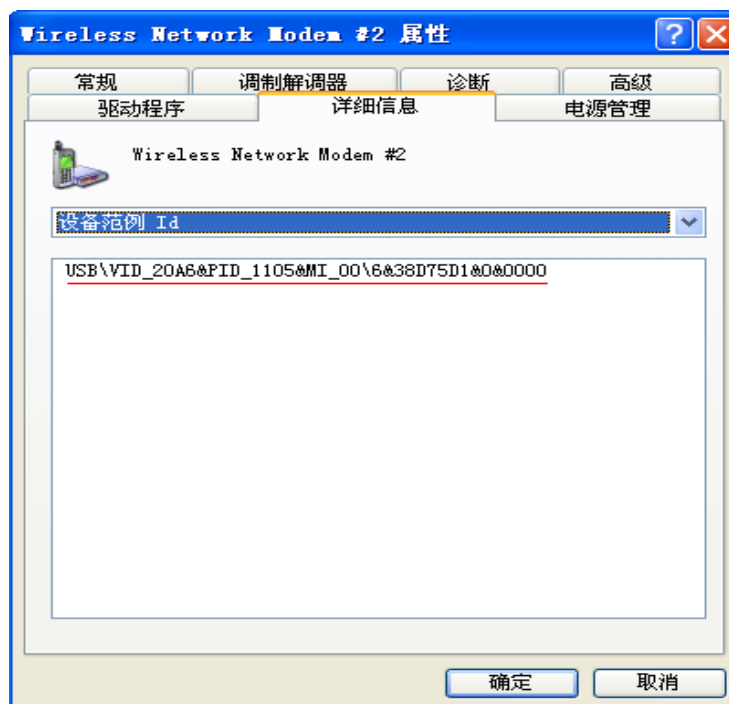
打开设备管理器，展开“调制解调器”和“端口(COM 和 LPT)”如下图所示:



确定 PPP 口



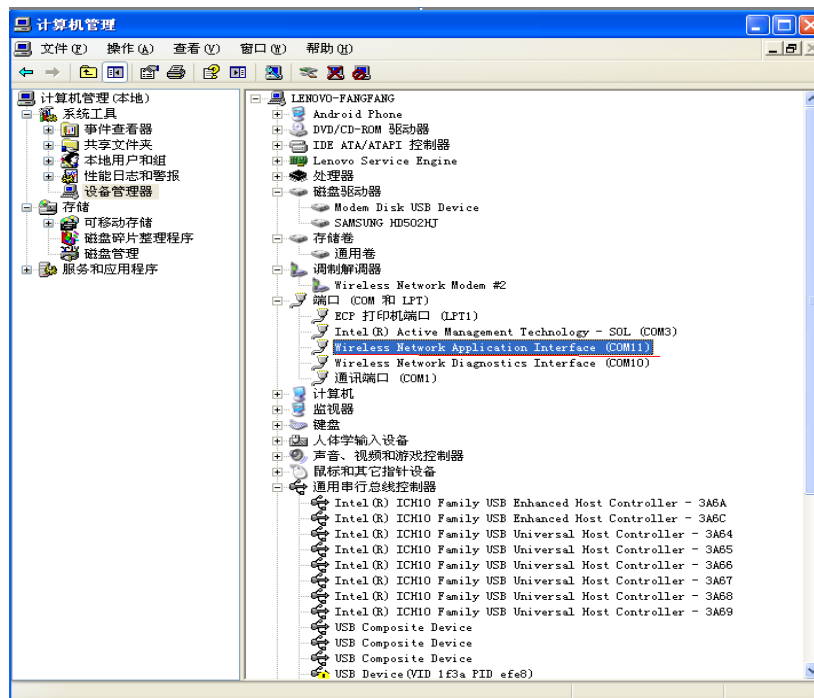
选择上图红色 Modem 口，单击右键，选择“属性”，然后点击“详细信息”，得到下图。



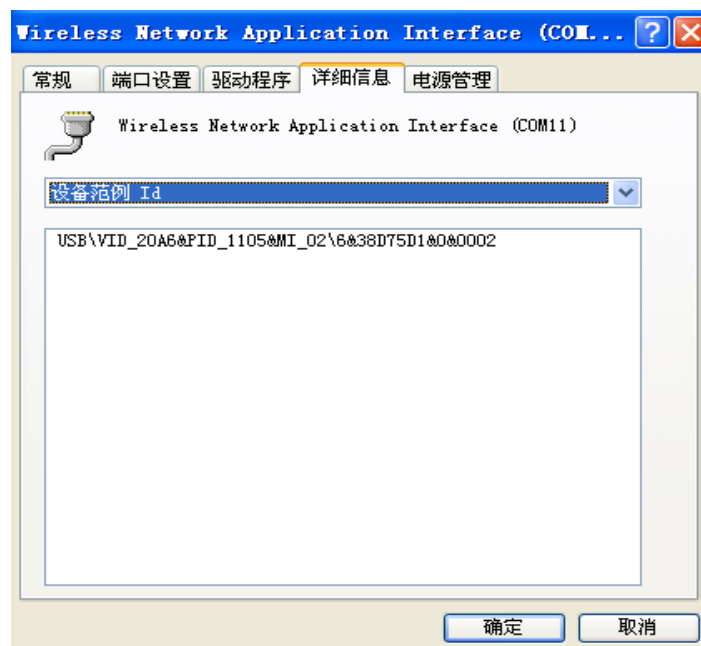
由 `USB\VID_20A6&PID_1105&MI_00\6&38D75D1&0&0000` 可以得到 vid,pid 和 ttyUSBx 中的 x。  
由 MI\_00 可以知道该端口在 Linux 系统对应的串口是 ttyUSB0, 故这个设备的 PPP 口是 ttyUSB0, 即 PPP 口为 0 口。

### 确定 AT 口

选择端口下面的设备, 这里选择带 “Application” 字样的串口。如下图所示:



同样的, 单击右键, 选择 “属性”, 然后单击 “详细信息”, 得到下图:

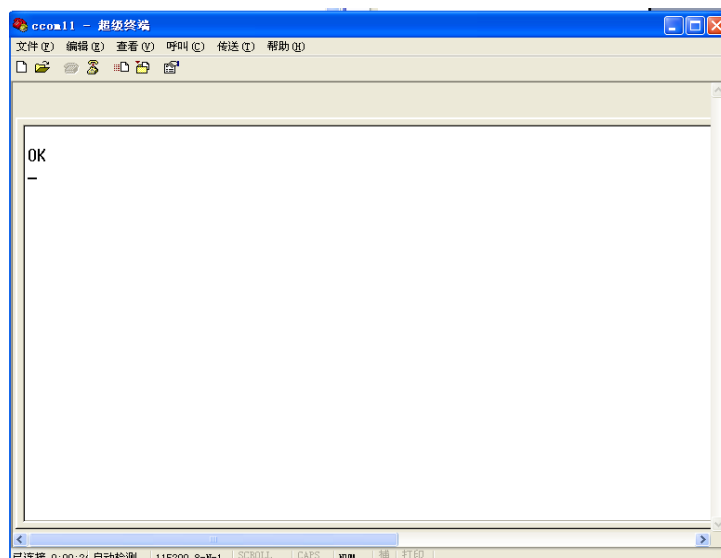


由 `USB\VID_20A6&PID_1105&MI_02\6&38D75D1&0&0002` 中的 MI\_02 得知 AT 口在 Linux 系统将是 `ttyUSB2`，这个信息在 `ril` 库代码修改中会用到。

这里会遇到的问题就是：有几个串口，而且从名字看不知道哪个口是 Application Interface（即 AT 口）。AT 口的确定需要超级终端或者其它串口工具来发送命令“AT”来测试。这里用超级终端来测试，这里选择 COM11，该串口的设置如下：



设置好串口后，发送指令“AT”测试，敲入‘AT’，按回车，得到‘OK’。如下所示：





**注意：**这里若提示打不开设备，则可能是你没有关闭该 3G Dongle 自带的应用程序。

上面只是一种参考方法，不一定能准确确定 at, ppp 口对应的 ttyUSB 口。这时候可以借助 Linux 或者 Android 下的串口工具来确定了。一般 at 和 ppp 可以发生 'AT' 命令，如果 ttyUSB0, ttyUSB1 两个口可以发 'AT' 命令，那么 at, ppp 口的组合也就两种：

(at=ttyUSB0, ppp=ttyUSB1) 和 (at=ttyUSB1, ppp=ttyUSB0)

## 4. 修改内核层

得到 TargetVendor=0x20a6, TargetProduct= 0x1105 后，需要在 option.c 中的数组 option\_ids 添加 vid, pid 信息。修改如下：

先做如下声明：

```
/* PuChuang WCDMA */
```

```
#define PUCHUANG_VENDOR_ID      0x20A6
```

```
#define PUCHUANG_PRODUCT_E003   0x1105 //wcdma
```

然后在数组 **option\_ids** 中添加 vid, pid 信息：

```
static const struct usb_device_id option_ids[] =
```

```
{
```

```
...
```

```
{USB_DEVICE(PUCHUANG_VENDOR_ID, PUCHUANG_PRODUCT_E003)},
```

```
...
```

```
}
```

这里如果没有 TargetVendor, TargetProduct, 则用 DefaultVendor, DefaultProduct 代替。

在 option.c 中添加 vid, pid 后，重新编译 linux 内核，这样的话，option 驱动才能支持这个新的 USB 设备。编译 linux 内核后，还得重新编译 android. 烧录新的 image.

Android 启动后把上面得到的 vid\_pid 文件拷贝到 ./etc/usb\_modeswitch.d 目录下（也可在烧录前就放到 ./etc/usb\_modeswitch.d 中去）。

重新启动 Android 设备，插入该 3G Dongle, 执行命令：ls /dev, 应该就可以看到 ttyUSBx 设备了。



## 5. 修改 android 层

方法一：修改 reference-ril.c（如果没有拿到 ril 库源码，请采用方法二）

找到文件：./hardware/ril/reference-ril/reference-ril.c，然后找到数组：

```
static const struct modem_info modem_table[],
```

找到该数组后添加该 3G Dongle 的信息就可以了。针对本文档这款 3G Dongle 添加的信息如下：

```
/*----- 1.3 wcdma others -----*/
{"MU930 F5.5", "16", "Qualcomm", GSM_MODE, 0x05c61000, 0x0, 0x05c66000, 0x0, 1, 0},
{"WU830 F6.4", "+CGMM:MSM6290 S790", "Qualcomm", GSM_MODE, 0x05c61000, 0x0, 0x05c66000, 0x0, 1, 0},
{"E003", "E003", "PuChuang", GSM_MODE, 0x20a6f00e, 0x0, 0x20a61105, 0x0, 2, 0},
{"SRT-H800", "HSPA USB MODEM", "Shichuangxing", GSM_MODE, 0x1e89f000, 0x0, 0x1e891a20, 0x0, 1, 3},
{"", "", "TechFaith", GSM_MODE, 0x1d091010, 0x0, 0x1d091000, 0x0, 1, 0},
```

修改完 reference-ril.c 后，编译该库，改名为 liballwinner-ril.so，拷贝到./system/lib 中，重新启动系统，这款 3G Dongle 就能上网了。

方法二：修改./etc/3g\_dongle.cfg

如何添加 3G USB Dongle 信息，请参考 3g\_dongle.cfg。

## 6. 常见问题

1.打印一直显示每个两秒钟寻找一次 USB 设备

- 1) 没有添加转换信息导致 3G Dongle 没有转换成功。
- 2) 添加了转换信息，但是没在 option.c 中添加设备信息，导致无法在 dev 目录下发现 USB 串口设备。

2.打印信息循环显示循环 8 秒去尝试打开 AT 口

Android 层没有添加转换后的 3G Dongle 信息(reference-ril.c 或 3g\_dongle.cfg)

3.打印信息显示不停的循环拨号

PPP 口或这 AT 口不对



## 7. Declaration

This **A10 android4.0 doc** is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.